RecvMsg

Carefully manage buffer size and ensure remote host is validated

Sean Barnum, Cigital, Inc. [vita[1]]

2007-04-02

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 6303 bytes

Attack Category	• Malicious Input • Identity Spoofing
Vulnerability Category	• Buffer Overflow
Software Context	• Networking
Location	• sys/socket.h
Description	recvmsg() is a socket operation which can operate on unconnected sockets. Calling recvmsg() will return a message (populated into a scatter/gather array) or an error. recvmsg() has two problems of which a programmer should be aware. First, the data is read off the wire into a buffer until the buffer is full or until the whole message has been read. The size of this buffer is determined by the iov_len element of the iovec struct. If this length is longer than the amount of memory allocated to the iov_base element of the iovec stuct then a buffer overflow condition could occur. The second pitfall is that recvmsg(), if the socket is not connection-oriented, will accept data from any remote host. When a message is received, if the address of the remote host is not validated against where data is expected to originate, a malicious remote host could inject arbitrary data. If this data is properly validated upon receipt, unexpected application behavior could result.
APIs	<table><tr><th>Function Name</th><th>Comments</th></tr><tr><td>recvmsg</td><td></td></tr></table>
Method of Attack	An attacker would first have to determine what host address and port from which the socket is reading data. If the host is not properly validating the origin of data, the attacker could send data from anywhere. Otherwise, the attacker would have to send data from a remote host from which the local host will accept

[1] http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

<table>
<tr><td></td><td>data. It should be noted that the source of the data could be forged.
Once an acceptable remote host is determined and if the size of the buffer is improperly-specified, a large enough message could be sent from (or appear to be sent from) this remote host to overflow the buffer. Experimentation might be required to determine how much data would be necessary to overflow the buffer.
Even if the buffer's length is properly specified abnormal program behavior could result if incoming data is not validated against expectations. Such behavior could range from no error at all to a security breach, depending on the role of the incoming data.</td></tr>
<tr><td><b>Exception Criteria</b></td><td>If the buffer length is properly-specified and if the remote host is properly confirmed and if the incoming data is properly validated, this function can be used safely.</td></tr>
<tr><td><b>Solutions</b></td><td>
<table>
<tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr>
<tr><td>This solution is applicable when alternatives to recvmsg() are available.</td><td>Consider using recv() and recvfrom() which are safer alternatives to recvmsg(). Both allow you to specify how much data you wish to receive (which should be less than or equal to the size of the buffer being used to receive the data). The former is connection-oriented and while the latter can be used in an unconnected context, it allows the specification of a remote host.</td><td>Proper use of recv() and recvfrom() (following many of the ideas outlined here) will eliminate the threat of a buffer overflow or use of unsanitized data.</td></tr>
<tr><td>This solution is applicable when one wishes to</td><td>Recvmsg() can be used safely as long as care</td><td>Using recvmsg() as outlined here</td></tr>
</table>
</td></tr>
</table>

	continue using recvmsg() but wants to make sure they are doing so in a prudent and secure manner.	is exercised. Set the size of the buffer as the same value used to specify how much memory would be allocated. Check the remote host address of any messages that are received as this will reduce the application's exposure to attack. Properly validate the incoming data, using regular expressions or other means, to confirm that it is how much data you expect.	will eliminate the threat of a buffer overflow or use of unsanitized data.
Signature Details	int recvmsg(int socket, struct msghdr * message, int flags)		

Examples of Incorrect Code	`/* Buffer size mismatch */`
	`struct msghdr message;`
	`//Allocate a buffer of length 10`
	`int bsize = 10;`
	`char buffer [bsize];`
	`message.msg_iov->iov_base = buffer;`
	`message.msg_iov->iov_len = 20; // WRONG SIZE!`
	`int res = recvmsg(sock, &message, 0);`		

Examples of Corrected Code	`/* Buffer correctly sized */`
	`struct msghdr message;`
	`//Allocate a buffer of length 10`
	`int bsize = 10;`
	`char buffer [bsize];`
	`message.msg_iov->iov_base = buffer;`
	`message.msg_iov->iov_len = bsize;`
	`int res = recvmsg(sock, &message, 0);`		

Source References	- http://www.hmug.org/man/2/recvmsg.php		

<table>
<tr><td></td><td>• http://www.gnu.org/software/libc/manual/html_node/Scatter_002dGather.html
• http://rootr.net/man/man/recv/2
• Rough Auditing Tool for Security (RATS)[5]</td></tr>
<tr><td><b>Recommended Resource</b></td><td></td></tr>
<tr><td><b>Discriminant Set</b></td><td><table><tr><td><b>Operating System</b></td><td>• UNIX (All)</td></tr><tr><td><b>Languages</b></td><td>• C
• C++</td></tr></table></td></tr>
</table>

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

[1] mailto:copyright@cigital.com